Advanced Disc Toolkit

16k EPROM

containing over 30 commands

A highly versatile toolkit from:-

ADVANCED COMPUTER PRODUCTS LIMITED

Suitable for:-

- BBC B
- BBC B+
- ELECTRON
- MASTER SERIES
- DFS
- 1770 DFS
- ADFS
- SECOND PROCESSOR

ADVANCED DISC TOOLKIT

CONTENTS

1. INTRODUCTION

2.A.D.T Commands

BACKUP
BFIND
BUILD
CATALL
DCOMP
DEX
DFIND
DIRALL
DUMP
ENVELOPE
FCOMP
FCOPY
FORM
FREE
FSN
KEYL
LIST
MAP
MDUMP
MENU
MEX
MFIND
MLOAD
MOVE
MRUN
ROMS
SECTORS
SETADR
SPT
SWAP
TYPE
UNPLUG
VERIFY
XFER

3. FITTING A.D.T

4. COMMAND SUMMARY

1. INTRODUCTION

A.D.T is a utility ROM designed to use the BBC computer or BBC 'Electron' fitted with the Disc Filing System DFS or the Advanced Disc Filing System ADFS.

It increases the computing power of the machine by adding new commands to the machine's operating system, MOS. These commands are instantly accessible from inside a program or typed in from the keyboard.

22 of the commands are disc utilities which use one of the Acorn

disc filing systems. The other commands are general utilities which can be used in any filing system, such as a TAPE machine without a disc interface.

A.D.T has 5 utilities which are also in DFS these are:-

BACKUP
BUILD
DUMP
LIST
TYPE

These are in A.D.T for machines which only have ADFS as a 'Disc Filing System'. This is because ADFS does not have these in ROM, they are supplied as utilities on disc. For information on the disc filing system, refer to the 'filing system's' Disc User Guide.

Chapter 2 explains the parameters used by ADT commands, followed by a description of each command, under the headings:

Purpose
Examples
Note

It is advisable not to just read the examples but to use them as well. This will often make the description of the command much clearer. In the example, text appearing like this should be typed into the computer.

Chapter 3 shows how to fit ADT into a BBC computer or Electron

Chapter 4 gives a brief description of each command.

2. ADT COMMANDS

Type *HELP ADT

This shows a list of all the commands. Their parameters are shown in brackets after the command. A parameter is a number, name, letter or address which the command needs to operate correctly. Some parameters are optional, meaning that the command will supply its own number or address. Optional parameters are shown in round brackets, e.g. (<start>), (<rom>), (<drive>), etc.

If a command is typed in with a parameter missing, a message will be displayed showing the correct syntax for the command, eg.
SYNTAX : MOVE <start> <end> <dest> (<rom>).

To avoid command names conflicting with other ROMs, ADT commands can be preceded with letter 'A' e.g. *AMENU is the same as *MENU and *AFORM is the same as *FORM. For instance, DFS and ADT both have the facility *DUMP. If you wanted to be sure of using the *DUMP in ADT then you should use *ADUMP.

Commands can be abbreviated by adding a full-stop to the abbreviated name so long as the name remains unique to ADT and other MOS commands. *XF. is the same as *XFER and *UN. is the same as *UNPLUG. *KEY is an MOS command and *KEYL is an ADT command. *KEYL cannot be abbreviated because it could be interpreted as *KEY by the MOS.

ADT commands will be accepted in both upper and lower case.

From now on, the BBC computer used with the 2nd processor will be referred to as the input/output processor.

Parameter Definitions:-

In this manual, the ampersand (&) printed in front of a number means that the number is in hexadecimal (base 16) otherwise the number is in decimal (base 10). When entering a Hex number in an ADT command '&' should NOT be printed in front of the number.

<b> Bytes, zero to 225.

Determines how data is printed on the screen and how much on each line about to be printed. A byte can be printed as a Hex number, an ASCI character, or part of a dissassembled line. The list below shows how a byte is printed and the byte increments per line for different values of <b>

<b>=0	6502 Dissassembler
<b>=1-99	ASCII plus HEX in <b> byte increments
<b>=100	ASCII in 32 byte increments
<b>=101-199	ASCII in <b>-100 byte increments
<b>=200	HEX in 11 byte increments
<b>=201-255	HEX in <b>-200 byte increments

<d.adr> Disc sector address

A sector address is a HEX number which refers to one sector on a disc. The *INFO command gives the location of a file as a disc sector address. To translate a track sector address into a disc sector address and vice versa, use the following formula:

For DFS:- Sector address=Track*10+sector
Track=Sector address DIV 10
Sector=Sector address MOD 10

For ADFS:-Sector address=Track*16+sector
Track=Sector address DIV 16
Sector=Sector address MOD 16

<dest> Destination 'drive' number

This is used by *MOVE to mean a destination memory address

<end> The 'end' memory address....see <start

Sometimes it is easier to express an 'end' memory address as the number of HEX bytes from the 'start' memory address. This can be done by preceding the number with '+' e.g +100,+500, etc.

<fs.afsp> Filing system name,Ambiguous file specification

<fsp> File specification

<load> Load memory address

(L) Large disc i.e. Double sided 80 track (ADFS only)

(M) Medium size disc i.e. Single sided 80 track (ADFS only)

<n> Decimal number 0 to 255

<rom> Sideways ROM number 0 to 15

<spt> Sectors per track 10 to 20

If the 'Acorn' DFS or ADFS is the current filing system then this parameter need not be used. It is assumed that DFS will use 10

sectors per track and ADFS 16.

Some DFS's use 16,17 or 18 sectors per track. In a double density mode 'spt' can be used to change the default number of sectors per track.

Note:- This parameter relies on the fact that the DFS supports a call to access any sector on a track via OSWORD &7F

<srce> Source drive number

<start> Start memory address

A memory address is a HEX number 0 to &FFFF FFFF, so called 32 bit addressing.

If a 2nd processor is not connected, addresses 0 to &FFFF will refer to memory in the I/O processor. The top 2 bytes are ed e.g. address &FFFF1234 is the same as &1234.

If the 2nd processor IS connected, addresses 0 to &FFFFFFFF will refer to memory in the 2nd processor, and addresses &FFFF0000 to &FFFFFFFF refer to memory in the I/O processor. To summarise:-

I/O processor only

addresses 0 to &FFFF	-I/O processor memory.

I/O processor & 2nd processor

addresses 0 to &FFFFFFFF	- 2nd processor memory
addresses &FFFF0000 to &FFFFFFFF	-I/O processor memory

For more information on the I/O and 2nd processor memory addresses refer to Chapter 9, 'Distinguishing between Memories' in the 6502 User Guide

<str> String

Strings which contain spaces, must be enclosed between double quotation marks. The 'wild card' character '#' can be used to replace a single character in a string to represent all possible characters e.g. AB# could represent ABA ABB ABC etc.

To enter a string as a HEX number precede the number with '&' e.g. &20F0FF.

(T) Tube

This parameter is only effective if the 2nd processor is connected to the I/O processor and turned 'On'.

Because the 2nd processor has more free memory than the I/O processor, which is not affected in size by screen mode or ROM software increasing the value of PAGE, commands which use the (T) option can make use of this extra memory. However, anything in memory before the command is used will be overwritten. The main advantage of using the 2nd processor is in increasing speed

*BACKUP <srce> <dest> (T)

Purpose:-

To copy the contents of one disc on to another. <srce> is the source drive from which sectors are read and <dest> is the destination drive to which the sectors

are written. If (T) is given then the 2nd processor will be used.

A disc can be copied using a single drive where <srce> and <dest> are the same number, but swapping the discs between reading & writing will be necessary.

Before copying the disc, a message is displayed to confirm that this is really what you want. Press 'Y' to copy the disc, or any other key to exit the command.

Examples

```
*BACKUP 0 1
GO Y/N?  Y
```

Copies the disc in drive 0 onto the disc in drive 1.

```
*BACKUP 0 1 T
GO Y/N?  Y
```

Copies the disc in drive 0 onto the disc in drive 1 using the second processor.

Note:-

Because the command uses free memory, it is advisable to save your program first. It is not possible to copy between discs of different sizes, e.g. 40 track to 80 track or different formats. e.g. DFS to ADFS.This command can only be used in a DISC filing system.

*BFIND <str>

Purpose:-

To search a BASIC program for every occurrence of the string <str> and print the line number and string where it is found. The rest of the BASIC line is not listed

Examples

```
*BFIND "hello"
```

searches for "hello"

```
*BFIND P%
```

searches for integer variable P%

```
*BFIND JSR#######
```

searches for a string of ten characters starting with "JSR"

```
*BFIND &F4
```

searches for the HEX byte &F4 which is the token for the BASIC keyword REM Any BASIC keyword may be searched for if the token of the keyword is known. These are listed on pages 483, 484 of the 'BBC User Guide'

*BUILD <fsp>

Purpose:-

To create a text file called <fsp> consisting of characters typed from the keyboard for the purpose of being 'EXECuted' later.A line of numbers is printed before each new line of text. Termination of the entry of characters can be achieved by pressing ESCAPE at the start of a new line.

Example

```
*BUILD !BOOT
0001 MODE 3
0002 VDU19,0,4,0,0,0,0
0003 *AMENU $
0004 <ESCAPE>
```

Creates an EXEC file called '!BOOT'

*CATALL

Purpose

To list the filenames in the current directory and the directories which are a member of the current directory. The directory name is printed first, followed by the filenames. Directories are shown with a D after their name. The list is indented by one space for each level of directory entered. This command is of most use in ADFS which uses a hierarchal 'tree' structure of directories, that can be nested to numerous levels.

Example

```
*CATALL
$
!BOOT
menu
GAME     D
 BOARD    D
  board1
  board2
  board3
 ARCADE   D
  arcade1
  arcade2
  arcade3
LIBRARY  D
 utility1
 utility2
 utility3
```

Lists the filenames in each directory starting from the current directory, in this example $.

Note

In DFS this command will list the filenames in the current directory only, since DFS does not allow directories to be nested, ie. a directory cannot contain another directory.

*DCOMP <srce> <dest> (<n>)

Purpose:-

To compare the disc in drive <srce> with the disc in

drive <dest> sector for sector and print the sector address of the sectors which are not the same. The command is terminated after the first 8 different sectors are found. If (n) is used, the command will be terminated after the first (n) different sectors. If (n) is zero the ALL different sector addresses will be printed.

A single drive can be used to compare 2 discs, but <srce> & <dest> will be the same number and the discs will have to be swapped after each 'read'.

Examples:-

*DCOMP 0 1

Compares the discs in drive 0 and drive 1 and terminates after the first 8 different sectors

*DCOMP 0 1 1

Compares the discs in drives 0 and 1 and terminates after the first different sector

Note:-

Comparing discs uses free memory and so it is advisable to save your program before using this command. It is not possible to compare 2 discs of different sizes or format e.g. 40t & 80t OR DFS & ADFS. This command can only be used on a DISC filing system.

*DEX (<d.adr>) (<spt>)

Purpose:-

To examine and edit disc sectors <d.adr> is a sector to be edited, by default 0, <spt> is the number of sectors per track, by default 10for DFS and 16 for ADFS.

Each sector contains 256 bytes of which some or all can be displayed at once depending upon the screen mode. Modes 0 & 3 will display the whole sector. Moving the cursor is done with the 4 'cursor keys' Editing a byte is done by over-typing the byte from the keyboard. The new byte will appear in HEX inside the round brackets and as an ASCII character above the flashing cursor. To enter a number of HEX bytes press the COPY key. Round brackets will change to square brackets. Now only HEX values will be accepted i.e. 0-9 and A-F. Press COPY again to enter characters as ASCII. Moving forward or back a sector is by using the 4 SHIFT/cursor keys. If the sector has been altered, a message is displayed for saving the sector back on to disc. Press 'Y' to save the sector or any other key not to save. Press ESCAPE to exit this command.

Editing Keys:-

Cursor keys	for cursor movement
SHIFT/left	back 1 sector
SHIFT/right	forward 1 sector
SHIFT/down	forward 1 track
SHIFT/ up	back 1 track
COPY	input type HEX or ASCII
CTRL/B	change background colour

CTRL/F change foreground colour
CTRL/P Print screen
ESCAPE Exit command

Examples:-

*DEX

To edit the first sector of the disc in the current drive

*DEX:1.30

To edit sector &30 of the disc in drive 1.

*INFO!BOOT
$!BOOT 000000 FFFFFF 000010 127

This shows that the !boot file starts at sector 127

*DEX 127
To edit the first sector of !BOOT

Note:-

A disc to be edited must be of the correct format for the current filing system, eg. If ADFS is the current filing system, then this command can only edit ADFS discs. The current filing system can be identified with the *FSN command. Editing discs is only possible in a Disc Filing System and in a screen mode which has 40 or 80 columns.

*DFIND <str> (<d.adr>) (<d.adr>)

Purpose:-

To search a disc for every occurrence of the string <str> and print the sector, byte address, and string where the string is found. The second parameter <d.adr> determines at which drive and sector the search will start, by default 0 in the current drive. The last parameter <d.adr> is the last sector, where the search will end

Examples:-

*DFIND !BOOT

Searches ALL sectors in the current drive for !BOOT

*DFIND"!BOOT $" 0 2

Searches sectors 0 and 1 in the current drive for !BOOT
*DFIND #BOOT :1.0 2

Searches the first 2 sectors in drive 1 for a 5 character string, ending in BOOT

*DFIND &214241:1
Searches the disc in drive 1 for the sequence of bytes 21 42 41

*DFIND ########

Displays all sectors in the current drive as ASCII

characters.

Note:-

This command can only be used in a disc filing system.

*DIRALL

Purpose

To list all the directory names in the current directory and the directories which are a member of the current directory. The list is indented by one space for each level of directory entered. This command is of most use in ADFS which uses a hierarchical tree of directories that can be nested.

Examples:-

```
*DIRALL
$
GAMES    D
 ARCADE   D
 BOARD    D
LIBRARY  D
```

Lists all directory names starting from the current directory, in this example $.

Note:-

In DFS this command will print the current directory name only since DFS does not allow directories to be nested.ie a directory cannot contain another directory.

*DUMP <fsp> (<b>)

Purpose:-

To display the contents of file <fsp>. <b> determines the format of the display, by default 8, which displays the file as hex bytes and ASCII characters in increments of 8 bytes.

Examples

*DUMP !BOOT

Displays !BOOT in HEX and ASCII

*DUMP MCODE 0

Displays MCODE dissassembled

*DUMP Letter 100

Displays Letter in ASCII only

*DUMP Numbers 200

Displays Numbers in HEX only

NOTE:-

Because DFS has a similar utility *DUMP this command my appear not to be working properly. To be sure that the ADT command is used, precede it with the letter 'A' e.g. *ADUMP

*ENVELOPE (<n>)

Purpose:-

To list ENVELOPE definition <n>, by default Envelope definitions 1-16. An explanation of the ENVELOPE parameters can be found on pages 182 & 245 of the 'USER GUIDE'

Example

*ENVELOPE

Lists the ENVELOPE definitions 1-16

*ENVELOPE 1

Lists definition 1

*ENVELOPE 1 2 3

List definitions 1,2 & 3

*FCOMP <fsp> <fsp> (<n>)

Purpose:-

To compare file<fsp> with file<fsp> byte for byte, and print the address and bytes from the 2 files in HEX and ASCII where they differ. The command is terminated after the first 8 differences are found. If <n> is used, the command terminates after the first <n> differences. If <n>=0 ALL differences will be printed.

Examples:

*FCOMP Letter Letter2

Compares Letter with Letter2 and terminates after the first 8 different bytes

*FCOMP Letter Letter2 1

Compares Letter with Letter2 and terminates after the first difference.

Note:-

A comparison of 2 files is only possible if they are the same length.

*FCOPY <fsp> <fsp< (T)

Purpose:-

To create a copy the file <fsp> and give it a new name <fsp>. The file names must be different. If (T) is given then the 2nd processor will be used. This is useful for copying large files.

Examples:

*FCOPY Letter Letter2

Creates a copy of Letter called Letter2

*FCOPY:0.Letter :1.Letter2

Creates a copy of Letter from drive 0, names it Letter2, and stores it on drive 1

*FORM 40/80 (<drive>) (M) (L) (C) (T)

Purpose:-

To initialize a new disc for reading and writing. The first parameter is the number of tracks to be formatted onto the disc. A 40 track drive will use 40 track discs and and 80 track drive will use 80 track discs. <drive> is the number which contains the disc to be formatted. (M) and (L) only apply to formatting ADFS discs. Use (M) to format a single sided 80 track disc and (L) to format a double sided 80 track disc. (C) only applies to formatting a DFS disc. It is used to create a dual catalogue disc.

The (T) parameter can be used to format a single track. This can be useful when a track has become damaged and formatting the whole disc would lose valuable data. To format a track use track number to be formatted instead of the 40/80 parameter and use (T) after the drive parameter.

The second and third parameters <drive> (M) (L) (C) can be repeated in the command to format more than one disc in the same command. Before formatting the first disc, a message is displayed to confirm that this is really what you want. Press 'Y' to format the disc of any other key to exit the command.

A track No. is printed in HEX as each track is formatted. After formatting a track it is verified for legibility. see *VERIFY.

Examples for DFS

*FORM 80
Format which drive ? 0
Go (Y/N) ? Y

Formats an 80 track disc in drive 0

*FORM 80 0 C 2 C
Go (Y/M) ? Y

Formats both sides of an 80 track disc in drive 0 with dual catalogues

Examples for ADFS

*FORM 80
Format which drive ? 0 (M) (L) ? M
Go (Y/N) ? Y

Formats a single sided 80 track disc in drive 0

*FORM 40 0 1

Go (Y/N) ? Y

Formats a single sided 40 track discs in drives 0 AND 1. It is not possible to format a double sided 40 track disc.

Note:-

DFS and ADFS use discs formatted to a different specification. The *FORM command will format a disc for use in the correct filing system e.g. It is not possible to format a DFS disc if the current filing system is ADFS. Use *FSN command to identify the current filing system. Formatting an ADFS disc uses free memory so it is advisable to save any program first.

Formatting an ADFS disc can cause parts of the screen to be overwritten in some modes. However this will not affect the formatting command

Dual Catalogue discs:

Discs formatted with a dual catalogue, divide the disc space into two separate catalogues of which only one can be used at a time. Each catalogue uses one half of the disc space. The advantage of this, is to have an extra 29 files per side of a disc, totalling 60. The disadvantage is that the maximum file size is halved. Each catalogue is formatted with a special file !.!!!!!!!! The files themselves are empty, it is the entry in the catalogue that is important. These files must not be deleted. Use *SWAP to swap the 2 catalogues.

*FREE (<drive>)

Purpose

To display the number of free and used files and disc space remaining and used on the disc in drive <drive>, by default the current drive. Free and used files are given in decimal. Disc space is given as sectors in hex and bytes in decimal.

Examples

*FREE

Displays the free space on the current drive.

*FREE 2

Displays the free space on drive 2.

Note

This command only operates in DFS on a DFS disc. However, ADFS has a similar command *FREE, which displays the amount of free space in the current drive.

*FSN (<n>)

Purpose:-

To identify the current filing system by name. If <n> is used, the filing system name, which has the filing system number <n> will be printed.e.g

Examples

```
*FSN
Disc Filing System
```

DFS is the current Filing System

```
*FSN 3
ROM filing system
```

Filing system number 3 is the Rom filing system.

*KEYL (<n>)

Purpose:-

To list function key definition <n>. By default definitions 0-15. A definition includes the *KEY syntax and key number, so it can easily be edited using the cursor and COPY keys. Function keys with no definitions are not listed.

Examples

*KEYL

Lists key definitions 0 to 15

*KEYL 5

Lists key definition 5

*KEYL 0 1

Lists key definitions 0 and 1

*LIST <fsp>

Purpose:-

To list the text file <fsp> with line numbers

Example

```
*LIST !BOOT
0000 MODE 3.
0001 VDU19,0,4,0,0,0,0
0002 *AMENU $
```

Lists the the text file !BOOT

*MAP (<drive>)

Purpose:-

To list a map of the free space on the disc in drive <drive>, by default the current drive. The map is printed as a list of disc addresses and lengths of free space in sectors both in HEX.

The free space map changes whenever a file is saved or deleted. This causes spaces to form between files,

fragmenting the disc space. To eliminate the free spaces, use the filing system's *COMPACT command.

Examples

*MAP

Lists the free space map on the current drive.

*MAP 1

Lists the free space on drive 1.

Note:-

This command only operates in DFS on a DFS disc. However ADFS has a similar command *MAP which lists the free space on the current drive.

*MDUMP <start> <end> (<b>) (<rom>)

Purpose:-

To display the contents of memory from <start> to <end>. <b> determines the format of the display, by default 8, which displays memory as HEX bytes and ASCII characters in increments of 8 bytes. (rom> determines which ROM is used if memory is read from addresses addresses &8000 to &BFFF in the I/0 processor.

Examples:

*MDUMP 1900 1A00

Displays memory from &1900 to &1A00 in HEX and ASCII

*MDUMP 8000+4000 0 14

Disassembles memory in ROM 14 from &8000 to &C000

*MDUMP FFFF8000+4000 0 14

Disassembles memory in ROM 14 from &8000 to &C000 in the I/O processor. The same as the previous example, but with a 2nd processor.

MODE 0
*MDUMP 0 FFFF 172

Displays memory in MODE0 from 0 to &FFFF in ASCII

MODE 0
*MDUMP 0 FFFF 18

Displays memory in MODE 0 from 0 to &FFFF in HEX and ASCII

*MENU (<dir>) (<rom>)

Purpose:-

To display the files in directory <dir>, by default the current directory, so that one can be selected for execution. The title of the directory is printed at the top of the screen and the entry names are printed below. An entry pointer => can be moved using the 4

cursor keys to point to one of the entries on the screen. To execute a program, place the pointer opposite the file name and press RETURN. The programme will be loaded into memory and run. To LOAD a program only press 'L'. To enter directory '$' press '$'.

The programme is run according to the type of program it is, ie. BASIC, EXEC, or Machine Code using the *MRUN command. Alternatively press 'X' to EXECute a file or 'R' to RUN a file or 'H' to CHAIN a BASIC programme.

In ADFS the directory entry can be a directory name. A directory name is shown with a 'D' after it. To enter the directory, move the pointer opposite the directory name and press RETURN. To move to a directory's parent, ie. one level back, press 'A'. To move to a previous directory press 'P'. Using these keys it is possible to to enter ALL directories on a disc and consequently, find any programme.

This command can be used to load a file into a Sideways ROM socket if the 'load address' of the file is &FFFF8000. Moving the pointer opposite the file and pressing 'L' will load the file into RAM socket <rom> Pressing RETURN or 'R' will LOAD and RUN the program provided that it is in a language like 'VIEW' or 'VIEWSHEET'. IF <rom> is omitted, the program will be loaded into the first socket found to contain RAM, e.g. socket 13 on a B+ or 7 on a Master. Pressing a HEX number 0 to F will load the program into that socket number and then redisplay the menu.

Press ESCAPE to exit this command.

Key Definitions

Cursor keys	Pointer movement
RETURN	RUN program
$	Enter directory $
0 to F	LOAD program into S/W RAM
H	CHAIN BASIC program
L	LOAD program
R	RUN program
X	EXECute
CTRL/B	Change background colour
CTRL/F	Change foreground colour
ESCAPE	Exit command

ADFS Key Functions

^	Enter parent directory
&	Enter 'root' directory (same as $)
P	Enter previous directory

Examples:

*MENU

Displays files from the current directory

*MENU $

Files from directory $

*MENU $ 0

Displays files from directory $ and selects Sideways ROM socket 0 Used when a program is loaded into SWR after pressing RETURN

Note:-

It is not possible to use this command in screen modes 2 & 5

*MEX (<start>) (<b>) (<rom>)

Purpose

To examine and edit the contents of memory. <start> is the start address from which memory is displayed, by default PAGE. <b> determines the format of the display, by default 8, which displays memory as hex bytes and ascii characters, in increments of 8 bytes. <rom> determines which Rom is used if memory is read from addresses &8000 to &BFFF in the I/O processor.

The cursor points to the current memory byte which is displayed as a HEX byte surrounded by round brackets, and an ASCII character above a flashing cursor. Moving the cursor is done by using the 4 cursor keys; editing is done by overtyping the current memory byte from the keyboard. The display will be updated to reflect the change in memory.

To enter numbers as HEX bytes press COPY. The round brackets will change to square brackets. Now only HEX numbers will be accepted ie 0-9 and A-F. Press COPY again to enter characters in ASCII.

Moving forward or back a screen is done using SHIFT/cursor keys. To move the current byte to the top left of the window, press CTRL/^. Whilst viewing memory from a ROM press CTRL/R to view memory from the next ROM No. To examine memory on the other side of the tube, if the 2nd processor is connected, press CTRL/T.

If memory is viewed in disassembler, it is possible to follow the address of a JSR, JMP or branch instruction by pressing RETURN whilst the memory pointer is over the instruction 'op/code'.

Press ESCAPE to exit this command.

Editing Keys

Cursor keys	Cursor movement
SHIFT/left cursor	Move cursor to left margin
SHIFT/right cursor	Move cursor to right margin
SHIFT/down cursor	Move 1 page down
SHIFT/up cursor	Move 1 page up
COPY	Swap cursor, HEX/ASCII
TAB	Cycle display format
CTRL/^	Home current memory byte
CTRL/B	Change background colour
CTRL/F	Change foreground colour
CTRL/P	Print screen
CTRL/R	Increment ROM No.
CTRL/T	Examine other side of 'Tube'
ESCAPE	Exit command

Disassembler Only

RETURN	Follow JSR,JMP or Branch
CTRL/X	Return from JSR
SPACE	Move cursor forward

Examples:

*MEX

To examine memory at PAGE in HEX and ASCII

*MEX 8000 0 14

To disassemble memory from &8000 in ROM 14

MODE 0
*MEX 8000 172 14

To examine memory in Mode 0 from &8000 in ROM 14 in ASCII

*MFIND <str> (<start>) (<end>) (<rom>)

Purpose:-

To search memory for the string <str> from <start> ,by default 0, to <end> by default &FFFF. <rom> is the ROM No. to read if the search includes memory from &8000 to &BFFF in the I/O processor. The address of the string in HEX and the string is printed for every occurrence of the string.

Examples:

*MFIND BASIC

Searches the memory for the string "BASIC"

*MFIND BASIC D000 DFFF

Searches from &D000 to &DFFF for the string "BASIC"

*MFIND &4241534943

Searches ALL memory for the bytes 42 41 53 49 43

*MFIND ################

Displays ALL memory in ASCII in increments of 16 bytes

Note:-

Using this command in screen mode 7 produces some surprising results when the search includes screen memory from &7C00 to &7FFF in the I/O processor. Every occurrence of the string is printed on the screen which in so doing produces another occurrence, rapidly filling the screen with strings. To solve this, confine the search to above or below screen memory or change to another screen mode.

*MLOAD <fsp> (<load>) (<rom>)

Purpose:-

To LOAD file <fsp> into memory and move it to the

address <load>. If <load> is omitted from the command, the file's LOAD address is used instead.

This command is useful for loading a program from disc which is designed to run at an address below PAGE. It can also be used to load a file into SWR socket <rom>. The file <fsp> is copied into SWR only if the address specified in the command is &FFFF8000 or &8000 if a 2nd processor is not connected. If <rom> is omitted from the command the file is loaded into the first socket found to contain RAM, e.g 13 on the B+ or 7 on the Master.

Examples:

*MLOAD GAME

Loads GAME and then moves it to its LOAD address

*MLOAD GAME E00

Loads GAME and moves it to &0E00

*MLOAD Romfile FFFF8000 0

Loads Romfile into SWR No.0

*MLOAD Romfile

Loads Romfile into SWR. The files load address must be FFFF8000. Use *SETADR to change a file's load address.

Note:-

If a program is moving to an address below OSHWM , normally PAGE, the Tape Filing System is selected. ROMs loaded into SWR which claim workspace like DFS should be initialized by pressing BREAK.

*MOVE <start> <end> <dest> (<rom>)

Purpose:-

To move a block of memory lying between <start> and <end> to the address <dest>. <rom> is the ROM No. used if <start> is an address between &8000 and &BFFF in the I/O processor.

Examples:

*MOVE 1900 1A00 1B00

Moves the block between 1900 and 1A00 to 1B00

*MOVE 8000+4000 2000 15

Moves the contents of ROM 15 to an address &2000

*MOVE FFFF8000+4000 800 15

Moves the contents of ROM 15 to address &800 If a 2nd processor was connected, memory would be copied from the I/O processor to the 2nd processor.

*MRUN <fsp> (<load>) (<exec>) (<rom>)

Purpose:-

To load file <fsp>, move it to the address <load>, and start execution of the program at address <exec>. IF <load> and/or <exec> are omitted from the command, the files load and/or execution addresses respectively are used instead. This command is useful for executing a BASIC or 'machine code' program that is designed to run at an address below PAGE.

This command can also be used to load and run a ROM file from SWR socket <rom> (see *MLOAD). The file <fsp> is loaded into SWR and run only if the program is a language like VIEW or VIEWSHEET.

This command makes a sensible guess about a program's language ie BASIC, EXEC or machine code and runs the program accordingly. It reads the file's execution address to determine its language. The list below shows which execution addresses can be used to identify a program. A file's address can be changed with the *SETADR command.

Language	Execution Address
BASIC	&8023
BASIC	&801F
BASIC	&B823
BASIC	&B82B
EXEC	&00000000
EXEC	&FFFFFFFF
ROM program	&FFFF8000
Machine code	any other address

Examples:

*MRUN GAME

Runs GAME using the file's load & execution addresses

Note:-

If a program is loaded below OSHWM the TAPE filing system is selected.

*ROMS (<rom>)

Purpose:-

To catalogue Sideways ROM <rom>, by default ROMS 0-15. <rom> can be the title of the ROM as an alternative to the ROM number. The catalogue of a ROM shows its ROM socket number, the type of program in the ROM, the title of the ROM and its version number if it has one. A ROM can be a language shown as (L), a service ROM (such as ADT) shown as (S) or both a language and a service ROM shown as (SL). A Rom which has been UNPLUGed will be shown as (**), see *UNPLUG.

Examples:

*ROMS

Catalogues ROMS 0-15

*ROMS 15

Catalogues ROM socket 15

*ROM BASIC

Catalogues the rom socket containing BASIC

*ROMS 12 13 14 15

Catalogues ROMS 12 to 15

*SECTORS <d.adr> <d.adr> <start> R/W

Purpose:-

To read or write sectors from or to a disc. The first parameter specifies the drive and the first sector to read or write. If a drive is not specified, the current drive is used. The second parameter specifies the last sector to read or write. <start> is the memory address to which the sectors are read or written. The last parameter, by default 'R', determines whether sectors are read from disc into memory 'R', or written to disc from memory 'W'.

The second parameter <d.adr> can be given as the number of bytes in HEX to read or write, if preceded by '+'. This makes it easy to read a file into memory using the files sector address and length.

Examples:

*SECTORS 0 2 1900

Reads sectors 0 and 1 on the current drive to &1900

*SECTORS :1.0 2 1900

Reads sectors 0 and 1 from drive 1 to address &1900

*INFO !BOOT
$.!BOOT 000000 FFFFFF 000010 027

Shows the first sector of !BOOT to be at sector &27

*SECTOR 27+10 1900

Reads !BOOT at sector &27 to address &1900

Note:-

This command can only be used in a Disc Filing System

*SETADR <fsp> <load> (<exec>)

Purpose:-

To set the load address of a file <fsp> to <load> and the execution address to <exec>. If <exec> is omitted, the file's execution address will remain unaltered. Use the filing system's *INFO command to display a file's addresses

Examples:

*SETADR GAME 1900

Changes GAME's load address to &1900
*SETADR GAME 1900 2300

Changes GAME's load address to &1900 and it's execution address to &2300

*SETADR GAME &1900 8023

Changes GAME's load address to &1900 and it's execution address to &8023. (The execution address &8023 is used by the *MRUN command to identify a BASIC program.)

*SPT (<n>)

Purpose:-

To change the default number of sectors per track to <n>. The default is 10 as used by 'Acorn' DFS. Some Acorn compatible DFS's can operate in a double density mode which use up to 18 sectors per track. This command can be used so that ADT commands like DEX, DFIND and SECTORS, will work on discs which have up to 18 sectors per track.

Example:

*SPT 18

Changes the number of sectors per track to 18

*SPT

Changes the number of sectors per track to 10

Note:-

This command can only be used in the Disc Filing System. Pressing BREAK re-sets the number of sectors per track to 10. This command will accept a decimal number between 1 and 31 inclusive. Any other number will be treated as 10.

*SWAP (<drive>)

Purpose:-

To swap the 2 catalogues on a dual catalogue disc in the current drive. See *FORM command.

Examples:

*SWAP

Swaps the catalogues on the disc in the current drive

*SWAP 1

Swaps the catalogues on the disc in drive 1.

Note:-

This command can only be used in the Disc Filing System on a dual catalogue DFS disc, formatted by the *FORM (C) command. All DFS command should operate normally on a dual catalogue disc, with the exception of *BACKUP.

It IS possible to backup a dual catalogue disc with the *BACKUP command provided the right catalogue on the disc, is the current catalogue. The 'right' catalogue is the one which has the most 'free' and 'used' space as displayed by the *FREE command. One of the 2 catalogues will have twice as much 'used' and 'free' space as the other. Ensure that this is the current catalogue before issuing a *BACKUP command.

*TYPE <fsp>

Purpose:-

To list text file <fsp> without line numbers. see *LIST

Example:

```
*TYPE !BOOT
MODE 3
VDU19,0,4,0,0,0,0
*AMENU $
```

Types the text file !BOOT

*UNPLUG (<rom>)

Purpose:-

To unplug SWR <rom>. This has the effect of turning a ROM off without having to physically remove it from it's socket. <rom> can be the title of the ROM as an alternative to the ROM No.

If <rom> is omitted from the command, each ROM title is displayed followed by :. Press 'Y' to UNPLUG or any other key to leave the ROM intact. If a ROM which has already been UNPLUGged is displayed and the response and the prompt is NO, the ROM will be plugged back IN.

Examples

*UNPLUG 14

To unplug the Rom in socket 14

*UNPLUG DFS

To unplug DFS. The value of PAGE will not be affected until BREAK or CTRL/BREAK is pressed.

Note:-

This command should be used with caution. Some ROMS can respond unpredictably to this command and not produce the intended effect. Pressing BREAK or CTRL/BREAK after UNPLUGing a ROM may, or may not, plug it back in. Type *HELP to see if the ROM has been plugged back in. To recover any lost ROM's, turn the machine off and then on, or use *FX200,2 and then press CTRL/BREAK.

*VERIFY <drive>

Purpose:-

To verify all sectors of a disc in drive <drive> for legibility. If <drive> is omitted from the command, a

message is displayed asking for the drive to verify. <drive> can be repeated in the command to verify more than one disc in the same command. A track number is printed in HEX as each track is verified.

If verifying a track fails on the first attempt, a '?' is printed after the track number. This could mean temporary corruption of the disc caused by foreign matter adhering to the surface of the disc.

If verifying a track, after 6 attempts the command is terminated and the disc fault is printed. This could mean permanent corruption of the disc.

Examples:

*VERIFY

Verifies the disc in drive 0

*VERIFY 0 1

Verifies the discs in drives 0 and 1

Note:-

This command will NOT overwrite a program in memory; it can only be used in a Disc Filing System.

*XFER <fs.afsp> <fs.afsp> (T)

Purpose:-

To copy a file from one filing system to another. The first parameter specifies the 'source filing system' and 'filename'. The second parameter specifies the 'destination filing system' and 'filename'. The filing systems should be given as a name e.g.TAPE, DISC, ADFS, NET. IF (T) is given, then the 2nd processor will be used. This is useful for transferring large files.

Examples:

*XFER TAPE DISC

To transfer ALL files on tape to the current directory in DFS. Filenames will be concatenated to 7 letters.

*XFER TAPE.MUSIC DISC

To transfer MUSIC from TAPE to the current directory in DFS

*XFER DISC.:1.$.MUSIC ADFS.:0.$.LIBRARY.SOUNDS

Transfers $.MUSIC in DFS on drive 1 to SOUNDS in directory LIBRARY in ADFS on drive 0

*XFER DISC ADFS T

Transfers all files in the current directory and drive in DFS to the current directory and drive in ADFS using the second processor

(Pencilled in note)
ADFS to DFS

```
*ADFS
*DIR :0.$
*DISC
*DIR :1.$
*XFER ADFS DISC (end of note)
```

Transferring files from ADFS to DFS

Insert the DFS disc into drive 0 and the ADFS disc into drive 1

```
*DISC
*DIR:0.$
*FADFS
*DIR :1.$. GAMES
*XFER DISC ADFS
```

This example would transfer all files in directory $ in DFS on drive 0 to directory $.GAMES in ADFS on drive 1.

Transferring files from DFS to ADFS using a single drive.

Insert a DFS disc into drive 0

```
*DISC
*DIR:0.$
*FADFS
*DIR :0.$.GAMES
*XFER DISC ADFS
```

This example will transfer files from directory $ in DFS on drive 0 to directory $.GAMES in ADFS on drive 0. Prompts are displayed when to swap the 2 discs. The 'source' disc is the one used with the source filing system and the destination is the one used with the destination filing system.

Note:-

This command will transfer files between filing systems which support the commands to read and write files. It is NOT possible to transfer a file from disc to ROM. Copying a large file from tape might cause the file to overwrite the screen in some modes. This might be avoided by selecting MODE 7 (MODE 6 on the ELECTRON) before transferring a file, or alternatively, using the (T) option if a 2nd processor is connected.

3. FITTING ADT

For the BBC computer

After turning off the power take off the top cover. Release the keyboard and fold it back to reveal the 4 ROM sockets in the bottom righthand corner. Insert the ADT into one of the empty ROM sockets, with the notch in the chip facing the rear of the computer. Take care not to bend any of the legs. Replace the keyboard and top cover.

4. COMMAND SUMMARY

```
*BACKUP   copy one disc to another
*BFIND    search a BASIC program for a string
*BUILD    create a text file
*CATALL   list filenames from the current directory
```

```
*DCOMP    compare 2 discs
*DEX      Disc sector editor
*DFIND    search a DISC for a string
*DIRALL   list directories from current directory
*DUMP     display the contents of a file
*ENVELOPE list the ENVELOPE definitions
*FCOMP    compare 2 files
*FCOPY    create a copy of a file
*FORM     initialize a new disc for reading & writing
*FREE     display the amount of free space on a disc
*FSN      identify the current filing system
*KEYL     list the function keys definitions
*LIST     list a text file with line numbers
*MAP      display a map of the free space on disc
*MDUMP    display the contents of memory
*MENU     select a program from a menu for execution
*MEX      memory editor
*MFIND    search MEMORY for a string
*MLOAD    load a program to a specific address
*MOVE     move a block of memory
*MRUN     run a program at a specific memory address
*ROMS     catalogue the SWR sockets
*SECTORS  read or write sectors onto a disc
*SETADR   change a file's LOAD and EXEC addresses
*SPT      change the default sectors per track
*SWAP     swap catalogues on a 'dual CAT' disc
*TYPE     list a text file WITHOUT line numbers
*UNPLUG   turn off a ROM
*VERIFY   verify a disc for legibility
*XFER     transfer files between 2 filing systems
```

Pencilled notes:-

```
Turn Off cursor VDU 23,1,0;0;0;0;
*COMPACT
TURN On  cursor VDU 23,2,1;0;0;0;
```